```
# Cancer Prediction Using Machine Learning

#Objective:
#To predict whether a tumor is Malignant (M) or Benign (B) using cell nucleus features.

#Tools:
#Python, Pandas, Scikit-learn, Google Colab

#Algorithm:
#Logistic Regression
```

```
import pandas as pd
```

```
cancer = pd.read_csv('https://github.com/YBIFoundation/Dataset/raw/main/Cancer.csv')
cancer.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1

5 rows × 33 columns

```
cancer.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_m
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1

5 rows × 33 columns

```
cancer.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

```
cancer.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mea
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.00000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.08879
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.07972
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.00000
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.02956
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.06154
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.13070
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.42680

8 rows × 32 columns

```
# Step 3 : define target (y) and features (X)
```

```
cancer.columns
```

```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

```
y = cancer['diagnosis']
```

```
X = cancer.drop(['id','diagnosis','Unnamed: 32'],axis=1)
```

```
# Step 4 : train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=2529)
```

```
# check shape of train and test sample
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((398, 30), (171, 30), (398,), (171,))
```

```
# Step 5 : select model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=5000)
```

```
# Step 6 : train or fit model
model.fit(X_train,y_train)
```

```
▾  LogisticRegression  i ?
LogisticRegression(max_iter=5000)
```

```
model.intercept_
```

```
array([-29.5754216])
```

```
model.coef_
```

```
array([[-0.90981805, -0.18594431,  0.24402858, -0.02468891,  0.14312991,
         0.1927918 ,  0.41547202,  0.22439816,  0.1759701 ,  0.03320957,
         0.0048194 , -1.19155032, -0.20208122,  0.12545286,  0.02428329,
        -0.05400143,  0.01489724,  0.02930236,  0.03092323, -0.01494273,
        -0.25755915,  0.44422964,  0.17881985,  0.01155521,  0.29404731,
         0.59263105,  1.05468156,  0.45535063,  0.53965235,  0.07607534]])
```

```
# Step 7 : predict model
y_pred = model.predict(X_test)
```

```
y_pred
```

```
array(['B', 'M', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'B',
       'M', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'B', 'B', 'M',
       'B', 'B', 'M', 'B', 'M', 'B', 'B', 'B', 'B', 'M', 'B', 'B', 'B',
       'M', 'M', 'M', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B',
       'B', 'B', 'B', 'B', 'B', 'M', 'M', 'M', 'B', 'M', 'B', 'M', 'M',
       'M', 'M', 'B', 'M', 'M', 'B', 'M', 'B', 'M', 'B', 'M', 'B', 'B',
       'M', 'M', 'M', 'B', 'B', 'M', 'M', 'M', 'B', 'B', 'B', 'B', 'M',
       'B', 'B', 'B', 'M', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'B', 'B',
       'B', 'M', 'B', 'B', 'M', 'B', 'B', 'B', 'M', 'B', 'B', 'B', 'B',
       'M', 'B', 'B', 'M', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'M',
       'M', 'B', 'M', 'M', 'M', 'B', 'B', 'M', 'B', 'M', 'B', 'M', 'B',
       'M', 'B', 'M', 'B', 'B', 'M', 'B', 'M', 'M', 'B', 'B', 'B', 'B',
       'B', 'M', 'M', 'M', 'M', 'B', 'B', 'B', 'M', 'B', 'M', 'B', 'B',
       'B', 'B'], dtype=object)
```

```
 #Step 8 : model accuracy
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[97,  5],
       [ 2, 67]])
```

```
accuracy_score(y_test,y_pred)
```

```
0.9590643274853801
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           B       0.98      0.95      0.97       102
           M       0.93      0.97      0.95        69

    accuracy                           0.96       171
   macro avg       0.96      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```